PART I
For Power Users

1

Darwinism: The Evolution of OS X

Mac OS has evolved tremendously since its inception. From a niche operating system of a cult crowd, it has slowly but surely gained mainstream share, with the recent years showing an explosion in popularity as Macbooks, Macbook Pros, and Airs become ever more ubiquitous, clawing back market share from the gradually declining PC. Its mobile derivative — iOS — is by some accounts the mobile operating system with the largest market share, head-to-head with Linux's derivative, Android.

The growth, however, did not happen overnight. In fact, it was a long and excruciating process, which saw Mac OS come close to extinction, before it was reborn as "OS X." Simply "reborn" is an understatement, as Mac OS underwent a total reincarnation, with its architecture torn down and rebuilt anew. Even then, Mac OS still faced significant hardship before the big breakthrough — which came with Apple's transition to Intel-based architecture, leaving behind its long history with PowerPC architectures.

The latest and greatest version, OS X 10.7, or Lion, occurred shortly before the release of this book, as did the release of iOS 5.*x*, the most recent version of iOS. To understand their features and the relationship between the two, however, it makes sense to take a few steps back and understand how the architecture unifying both came to be.

The following is by no means a complete listing of features, but rather a high-level perspective. Apple has been known to add hundreds of features between releases, mostly in GUI and application support frameworks. Rather, more emphasis is placed on design and engineering features. For a comprehensive treatise on Mac OS versions to date, see Amit Singh's work on the subject[1], or check Ars Technica's comprehensive reviews[2]. Wikipedia also maintains a fairly complete list of changes[3].

THE PRE-DARWIN ERA: MAC OS CLASSIC

Mac OS Classic is the name given the pre-OS X era of Mac OS. The operating system then was nothing much to boast about. True, it was novel in that it was an all-GUI system (earlier versions did not have a command line like today's "Terminal" app). Memory management was poor, however, and multitasking was cooperative, which — by today's standards — is considered primitive. Cooperative multitasking involves processes voluntarily yielding their CPU timeslice, and works reasonably well when processes are well behaved. If even one process refuses to cooperate, however, the entire system screeches to a halt. Nonetheless, Mac OS Classic laid some of the foundations for the contemporary Mac OS, or OS X. Primarily, those foundations include the "Finder" GUI, and the file system support for "forks" in the first generation HFS file system. These affect OS X to this very day.

THE PRODIGAL SON: NEXTSTEP

While Mac OS experienced its growing pains in the face of the gargantuan PC, its founder Steve Jobs left Apple (by some accounts was ousted) to get busy with a new and radically different company. The company, NeXT, manufactured specialized hardware, the NeXT computer and NeXTstation, with a dedicated operating system called NeXTSTEP.

NeXTSTEP boasted some avant-garde features for the time:

- NeXTSTEP was based on the Mach microkernel, a little-known kernel developed by Carnegie Mellon University (CMU). The concept of a microkernel was, itself, considered a novelty, and remains rarely implemented even today.
- The development language used was *Objective*-C, a superset of C, which — unlike C++ — is heavily object-oriented.
- The same object-orientation was prevalent all throughout the operating system. The system offered frameworks and kits, which allowed for rapid GUI development using a rich object library, based on the `NSObject`.
- The device driver environment was an object-oriented framework as well, known as DriverKit. Drivers could subclass other drivers, inheriting from them and extending their functionality.
- Applications and libraries were distributed in self-contained *bundles*. Bundles consisted of a fixed directory structure, which was used to package software, along with its dependencies and related files, so installing and uninstalling could be as easy as moving around a folder.
- *PostScript* was heavily used in the system, including a variant called "display postscript," which enabled the rendering of display images as postscript. Printing support was thus 1:1, unlike other operating systems, which needed to convert to a printer-friendly format.

NeXTSTEP went down the road of better operating systems (remember OS/2?), and is nowadays extinct, save for a GNUStep port. Yet, its legacy lives on to the present day. One winter day in 1997, Apple — with an OS that wasn't going anywhere — ended up acquiring NeXT, bringing its intellectual property into Apple, along with Steve Jobs. And the rest, as they say, is history.

ENTER: OS X

As a result of the acquisition of NeXT, Apple gained access to Mach, Objective-C, and the other aspects of the NeXTSTEP architecture. While NeXTSTEP was discontinued as a result, these components live on in OS X. In fact, OS X can be considered as a fusion of Mac OS Classic and

NeXTSTEP, mostly the latter absorbing the former. The transition wasn't immediate, and Mac OS passed through an interim operating system called Rhapsody, which never really went public. It was Rhapsody, however, that eventually evolved into the first version of Mac OS X, and its kernel became the core of what is now known as Darwin.

Mac OS X is closer in its design and implementation to NeXTSTEP than it is to any other operating system, including Apple's own OS 9. As you will see, the core components of OS X — Cocoa, Mach, IOKit, the XCode Interface Builder, and others — are all direct descendants of NeXTSTEP. The fusion of two fringe, niche operating systems — one with a great GUI and poor design, the other with great design but lackluster GUI — resulted in a new OS that has become far more popular than the both of them combined.

OS X VS. DARWIN

There is sometimes some confusion between OS X and Darwin regarding the definitions of the two terms, and the relationship between them. Let's attempt to clarify this:

OS X is the name given, collectively, to the entire operating system. As discussed in the next chapter, the operating system contains many components, of which Darwin is but one.

Darwin is the UNIX-like core of the operating system, which is itself comprised of the kernel, XNU (an acronym meaning "X is Not UNIX", similar to GNU's recursive acronym) and the runtime. Darwin is open source (save for its adaptation to ARM in iOS, discussed later), whereas other parts of OS X — Apple's frameworks — are not.

There exists a straightforward correlation between the version of OS X and the version of Darwin. With the exception of OS X 10.0, which utilized Darwin 1.3. *x*, all other versions follow a simple equation:

```
If (OSX.version == 10.x.y)
 Darwin.version = (4+x).y
```

So, for example, the upcoming Mountain Lion, being 10.8.0, is Darwin 12.0. The last release of Snow Leopard, 10.6.8, is Darwin 10.8. It's a little bit confusing, but at least it's consistent.

OS X VERSIONS, TO DATE

Since its inception, Mac OS X has gone through several versions. From a novel, but — by some accounts — immature operating system, it has transformed into the feature-rich platform that is Lion. The following section offers an overview of the major features, particularly those which involve architectural or kernel mode changes.

10.0 — Cheetah and the First Foray

Mac OS X 10.0, known as Cheetah, is the first public release of the OS X platform. About a year after a public beta, Kodiak, Apple released 10.0 in March 2001. It marks a significant departure from the old-style Mac OSes with the integration of features from NeXT/Openstep, and the layered architecture we will discuss shortly. It is a total rewrite of the MacOS 9, and shares little in common, save for maybe the Carbon interface, which is used to maintain compatibility with OS 9 APIs. 10.0 ran five sub-versions (10.0 through 10.0.4) with relatively minor modifications. The version of the core OS packages, called Darwin, were 1.3.1 in all. XNU was version 123.

10.1 — Puma — a Stronger Feline, but . . .

While definitely novel, OS 10.0 was considered to be immature and unstable, not to mention slow. Although it boasted preemptive multitasking and memory protection, like all its peer operating systems, it still left much to be desired. Some six months later, Mac OS X 10.1, known as Puma, was released to address stability and performance issues, as well as add more user experience features. This also led shortly thereafter to Apple's public abandonment of Mac OS 9, and focus on OS X as the new operating system of choice. Puma ran six sub-versions (10.1 through 10.1.5). In version 10.1.1, Darwin (the core OS) was renumbered from v1.4.1 to 5.1, and since then has followed the OS X numbers consistently by being four numbers ahead of the minor version, and aligning its own minor with the sub-version. XNU was version 201.

10.2 — Jaguar — Getting Better

A year later saw the introduction of Mac OS X 10.2, known as Jaguar, a far more mature OS with myriad UX feature enhancements, and the introduction of the "Quartz Extreme" framework for faster graphics. Another addition was Apple's Bonjour (then called Rendezvous), which is a form of ZeroConf, a uPNP-like protocol (Universal Plug and Play) allowing Apple devices to find one another on a local area network (discussed later in this book). Darwin was updated to 6.0. 10.2 ran nine sub-versions (10.2 through 10.2.8, Darwin 6.0 through 6.8, respectively). XNU was version 344.

10.3 — Panther and Safari

Yet another year passed, and in 2003 Apple released Mac OS X 10.3, Panther, enhancing the OS with yet more UX features such as Exposé. Apple created its own web browser, Safari, displacing Internet Explorer for Mac as it distanced itself from Microsoft.

Another noteworthy improvement in Panther is FileVault, which allows for transparent disk encryption. Mac OS X 10.3 stayed current for a year and a half, and ran 10 sub-versions (10.3 through 10.3.9) with Darwin 7.x (7.0 through 7.9). XNU was version 517.

10.4 — Tiger and Intel Transition

The next update to Mac OS was announced in May 2004, but it took almost a year until Mac OS X 10.4 (Tiger) was officially released. This version sported, as usual, many new GUI features, such as spotlight and dashboard widgets, but also significant architectural changes, most important of which was the foray into the Intel x86 processor space, with 10.4.4. Until that point, Mac OS required a PowerPC architecture. 10.4.4 was also the first OS to introduce the concept of *universal binaries* that could operate on both PPC and x86 architectures. The kernel was significantly improved, allowing for 64-bit pointers.

Other important developer features in this release included four important frameworks: Core Data, Image, Video, and Audio. Core Data handled data manipulation (undo/redo/save). Core Image and Core Video accelerated graphics by exploiting GPUs, and Core Audio built audio right into the OS — allowing for Mac's text-to-speech engine, Voice Over, and the legendary "say" command ("Isn't it nice to have a computer that talks to you?").

Tiger reigned for over two years and a dozen sub-versions — 10.4.0 (Darwin 8.0) through 10.4.11 (Darwin 8.11). XNU was 792.

10.5 — Leopard and UNIX

Leopard was over a year in the making. Announced in June 2006, but not released until October 2007, it boasted hundreds of new features. Chief among them from the developer perspective were:

- Core Animation, which offloaded animation tasks to the framework
- Objective-C 2.0
- OpenGL 2.1
- Improved scripting and new languages, including Python and Ruby
- Dtrace (ported from Solaris 10) and its GUI, Instruments
- FSEvents, allowing for Linux's inotify-like functionality (file system/directory notifications)
- Leopard is also fully UNIX/POSIX-compliant

Leopard ran 10.5 through 1.0.5.8; Darwin 9.0 through 9.8. XNU leapt forward to version 1228.

10.6 — Snow Leopard

Snow Leopard introduced quite a few changes, but mostly under the hood. Following what now was somewhat of a tradition, it took over a year from its announcement in June 2008 to its release in August 2009 From the UX perspective, changes are minimal, although all its applications were ported to 64-bit. The developer perspective, however, revealed significant changes, including:

- **Full 64-bit functionality**: Both in user space libraries and kernel space (K64).
- **File system–level compression**: Incorporated very quietly, as most commands and APIs still report the files' real sizes. In actuality, however, most files — specifically those of the OS — are transparently compressed to save disk space.
- **Grand Central Dispatch**: Enabled multi-core programming through a central API.
- **OpenCL**: Enabled the offloading of computations to the GPU, utilizing the ever-increasing computational power of graphics adapters for non-graphic tasks. Apple originally developed the standard, and still maintains the trademark over the name. Development has been handed over to the Khronos group (`www.khronos.org`), a consortium of industry leaders (including AMD, Intel, NVidia, and many others), who also host OpenGL (for graphics) and OpenSL (for sound).

Snow Leopard finished the process of migration started in 10.4.4 — from PPC to x86/x64 architectures. It no longer supports PowerPCs so universal binaries to support that architecture are no longer needed, saving much disk space by thinning down binaries. In practice, however, most binaries still contain multiple architectures for 32-bit and 64-bit Intel.

The most current version of Snow Leopard is 10.6.8 (Darwin 10.8.0), released July 2011. XNU is version 1504.

10.7 — Lion

Lion is Apple's latest incarnation of OS X at the time of this writing. (More accurately, the latest one publicly available, as Mountain Lion has been released as a developer preview as this book goes to print.) It is a relatively high-end system, requiring Intel Core 2 Duo or better to run on (although successfully virtualized by now).

While it provides many features, most of them are in user mode. Several of the new features have been heavily influenced from iOS (the mobile port of OS X for i-Devices, as we discuss later). These features include, to name but a few:

- **iCloud**: Apple's new cloud-based storage is tightly integrated into Lion, enabling applications to store documents in the cloud directly from the Objective-C runtime and NSDocument.
- **Tighter security**: Drawing on a model that was started in iOS, of application sandboxing and privilege separation.
- **Improvements in the built-in applications**: Such as Finder, Mail, and Preview, as well as porting of apps from iOS, notably FaceTime and the iOS-like LaunchPad.
- **Many framework features**: From overlay scrollbars and other GUI enhancements, through voice over, text auto-correction similar to iOS, to linguistic and part-of-speech tagging to enable Natural Language Processing–based applications.
- **Core Storage**: Allowing logical volume support, which can be used for new partitioning features. A particularly useful feature is extending file systems onto more than one partition.
- **FileVault 2**: Used for encryption of the filesystem, down to the root volume level — marking Apple's entry into the Full Disk Encryption (FDE) realm. This builds on Core Storage's encryption capabilities at the logical volume level. The encryption is AES-128 in XTS mode, which is especially optimized for hard drive encryption. (Both Core Storage and File Vault are discussed in Chapter 15 of this book, "Files and Filesystems.")
- **Air Drop**: Extends Apple's already formidable peer-finding abilities (courtesy of Bonjour) to allow for quick file sharing between hosts over WiFi.
- **64-bit mode**: Enabled by default on more Mac models. Snow Leopard already had a 64-bit kernel, but still booted 32-bit kernels on non-Pro Macbooks.

At the time of this writing, the most recent version of Lion is 10.7.3, XNU version 1699.24.23. With the announcement of Mountain Lion (destined to be 10.8), it seems that Lion will be especially short lived.

10.8 — Mountain Lion

In February 2012, just days before this book was finalized and sent off to print, Apple surprised the world with the announcement of OS X 10.8, Mountain Lion. This is quite unusual, as Apple's OS lifespan is usually longer a year, especially for a cat as big as a Lion, which many believed would end the feline species. The book makes every attempt to also include the most up-to-date material so as to cover Mountain Lion, but the operating system will only be available to the public much later, sometime around the summer of 2012.

Mountain Lion aims to bring iOS and OS X closer together, as was actually speculated in this book (see "The Future of OS X," later in this chapter). Continuing the trend set by Lion, 10.8 further brings features from iOS to OS X, as boasted by its tagline — "Inspired by iPad, reimagined for Mac." The features advertised by Apple are mostly user mode. Interestingly enough, however, the kernel seems to have undergone major revisions as well, as is hinted by its much higher version number — 2050. One notable feature is kernel address space randomization, a feature that is expected to make OS X far more resilient to rootkits and kernel exploitation. The kernel will also likely be 64-bit only, dropping support for 32-bit APIs. The sources for Darwin 12 (and, with them, XNU) will not be available until Mountain Lion is officially released.

Using uname(1)

Throughout this book, many UNIX and OS X-specific commands will be presented. It is only fitting that `uname(1)`, which shows the UNIX system name, be the first of them. Running `uname` will give you the details on the architecture, as well as the version information of Darwin. It has several switches, but `-a` effectively uses all of them. The following code snippets shownin Outputs 1-1a through c demonstrate using `uname` on two different OS X systems:

OUTPUT 1-1A: Using uname(1) to view Darwin version on Snow Leopard 10.6.8, a 32-bit system

```
morpheus@ergo (~) uname -a
Darwin Ergo 10.8.0 Darwin Kernel Version 10.8.0: Tue Jun  7 16:33:36 PDT 2011; root:xnu-
1504.15.3~1/RELEASE_I386 i386
```

OUTPUT 1-1B: Using uname(1) to view Darwin version on Lion 10.7.3, a 64-bit system

```
morpheus@Minion (~) uname -a
Darwin Minion.local 11.3.0 Darwin Kernel Version 11.3.0: Thu Jan 12 18:47:41 PST 2012;
root:xnu-1699.24.23~1/RELEASE_X86_64 x86_64
```

If you use `uname(1)` on Mountain Lion (in the example below, the Developer Preview) you will see an even newer version

OUTPUT 1-1C: Using uname(1) to view Darwin version on Mountain Lion 10.8 (DP3), a 64-bit system

```
morpheus@Simulacrum (~) uname -a
Darwin Simulacrum.local 12.0.0 Darwin Kernel Version 12.0.0: Sun Apr  8 21:22:58 PDT
2012; root:xnu-2050.3.19~1
```

OS X ON NON-APPLE HARDWARE

À la Apple, running OS X on any hardware other than the Apple line of Macs constitutes a violation of the EULA. Apple wages a holy war against Mac clones, and has sued (and won against) companies like Psystar, who have attempted to commercialize non-Apple ports of OS X. This has not deterred many an enthusiast, however, from trying to port OS X to the plain old PC, and — recently — to run under virtualization.

The OpenDarwin/PureDarwin projects take the open source Darwin environment and make of it a fully bootable and installable ISO image. This is carried further by the OSX86 project, which aims to fully port OS X onto PCs, laptops, and even netbooks (this is commonly referred to as "Hackintosh"). With the bootable ISO images, it is possible to circumvent the OS X installer protections and install the system on non-Apple hardware. The hackers (in the good sense of the word) emulate the EFI environment (which is the default on Mac hardware, but still scarce on PC) using a boot loader (Chameleon) based on Apple's Boot-132, which was a temporary boot loader used by Apple back in Tiger v10.4.8. Originally, some minor patches to the kernel were needed, as well — which were feasible since XNU remains open source.

With the rise of virtualization and the accessibility of excellent products such as VMWare, users can now simply download a pre-installed VM image of a fully functioning OS X system. The first images made available were of the later Leopards, and are hard to come by, but now images of the latest Lion and even Mountain Lion are readily downloadable from some sites.

While still in violation of the EULA, Apple does not seem as adamant (yet?) in pursuing the non-commercial ports. It has added features to Lion which require an Internet connection to install (i.e. "Verify the product with Apple"), but still don't manage to snuff the Hackintosh flame. Then again, what people do in the privacy of their own home is their business.

IOS — OS X GOES MOBILE

Windows has its Windows Mobile, Linux has Android, and OS X, too, has its own mobile derivative — the much hyped iOS. Originally dubbed iPhone OS (until mid-2010), Apple (following a short trademark dispute with Cisco), renamed the operating system iOS to reflect the unified nature of the operating system which powers all its i-Devices: the iPhone, iPod, iPad, and Apple TVs.

iOS, like OS X, also has its version history, with its current release at the time of writing being iOS 5.1. Though all versions have code names, they are private to Apple and are usually known only to the jailbreaking community.

1.*x* — Heavenly and the First iPhone

This release ran from the iPhone's inception, in mid-2007, through mid-2008. Version numbers were 1.0 through 1.02, then 1.1 through 1.1.5. The only device supported was initially the iPhone, but the iPod Touch soon followed. The original build was known as "Alpine" (which is also the default root password on i-Devices), but the released version was "Heavenly."

From the jailbreakers' perspective, this release was heavenly, indeed. Full of debug symbols, unencrypted, and straightforward to disassemble. Indeed, many versions later, many jailbreakers still rely on the symbols and function-call graphs extracted from this version.

2.*x* — App Store, 3G and Corporate Features

iPhoneOS 2.0 (known as BigBear) was released along with the iPhone 3G, and both became an instant hit. The OS boasted features meant to make the iPhone more compatible with corporate needs, such as VPN and Microsoft Exchange support. This OS also marked the iPhone going global, with support for a slew of other languages.

More importantly, with this release Apple introduced the App Store, which became the largest software distribution platform in the world, and helped generate even more revenue for Apple as a result of its commission model. (This is so successful that Apple has been trying this, with less success, with the Mac App Store, as of late Snow Leopard).

2.*x* ran 2.0–2.02, 2.1 (SugarBowl), 2.2–2.2.1 (Timberline), until early 2009, and the release of 3.*x*. The XNU version in 2.0.0 is 1228.6.76, corresponding to Darwin 9.3.1.

3.*x* — Farewell, 1st gen, Hello iPad

The 3.*x* versions of iOS brought along the much-longed-for cut/paste, support for lesser used languages, spotlight searches, and many other enhancements to the built-in apps. On the more technical front, it was the first iOS to allow tethering, and allowed the plugging in of Nike+ receivers, demonstrating that the i-Devices could not only be clients but hosts for add-on devices themselves.

3.0 (KirkWood) was quickly superseded by 3.1 (NorthStar), which ran until 3.1.3, the final version supported by the "first generation" devices. Version 3.2 (WildCat) was introduced in April of 2010, especially for the (then mocked) tablet called the iPad. After its web-based jailbreak by Comex (Star 2.0), it was patched to 3.2.2, which was its last version. The Darwin version in 3.1.2 was 10.0.0d3, and XNU was at 1357.5.30.

4.*x* — iPhone 4, Apple TV, and the iPad 2

The 4.*x* versions of iOS brought along many more features and apps, such as FaceTime and voice control, with 4.0 introduced in late June 2010, along with the iPhone 4. 4.*x* versions were the first to support true multitasking, although jailbroken 3.*x* offered a crude hack to that extent.

iOS 4 was the longest running of the iOS versions, going through 4.0–4.0.2 (Apex), 4.1 (Baker or Mohave, which was the first Apple TV version of iOS), and 4.2–4.2.10 (Jasper). Version 4.3

(Durango) brought support for the (by then well respected) iPad 2 and its new dual-core A5 chip. Another important new feature was Address Space Layout Randomization (ASLR, discussed later in this book), which was unnoticeable by users, but — Apple hoped — would prove insurmountable to hackers. Hopes aside, by version 4.3.3 ASLR succumbed to "Saffron" hack when jailbreaker Comex then released his ingenious "Star 3.0" jailbreak for the till-then-unbreakable iPad 2. Apple quickly released 4.3.4 to fix this bug (discussed later in this book as well), and figured the only way to discourage future jailbreaks is to go after the jailbreaker himself — assimilating him. The last release of 4.3.*x* was 4.3.5, which incorporated another minor security fix.

The Darwin version in 4.3.3 is 11.0.0, same as Lion. The XNU kernel, however, is at 1735.46.10 — way ahead of Lion.

5.*x* — To the iPhone 4S and Beyond

iOS is, at the time of this writing, in its fifth incarnation: Telluride (5.0.0 and 5.0.1) and Hoodoo (5.1), named after ski resorts. Initially released as iOS 5.0, it coincided with the iPhone 4S, and introduced (for that phone only) Apple's natural language-based voice control, Siri. iOS5 also boasts many new features, such as much requested notifications, NewsStand (an App Store for digital publications), and some features iOS users never knew they needed, like Twitter integration. Another major enhancement is iCloud (also supported in Lion).

As a result of complaints concerning poor battery life in 5.0, Apple rushed to release 5.0.1, although some complaints persisted. Version 5.1 was released March 2012, coinciding with the iPad 3.

As this book goes to print, the iPhone 4S is the latest and greatest model, and the iPad 3 has just been announced, boasting the improved A5X with quad-core graphics. If Apple's pattern repeats itself, it seems more than likely that it will be followed by the highly anticipated iPhone 5. Apple's upgrade cycles have, thus far, been first for iPad, then iPhone, and finally iPod. From the iOS perspective this matters fairly little — the device upgrades have traditionally focused on better hardware, and fairly few software feature enablers.

Darwin is still at 11.0.0, but XNU is even further ahead of Lion with the version being 1878.11.8 in iOS 5.1.

iOS vs. OS X

Deep down, iOS is really Mac OS X, but with some significant differences:

➤ The architecture for which the kernel and binaries are compiled is ARM-based, rather than Intel i386 or x86_64. The processors may be different (A4, A5, A5X, etc), but all are based on designs by ARM. The main advantage of ARM over Intel is in power management, which makes their processor designs attractive for mobile operating systems such as iOS, as well as its arch-nemesis, Android.

➤ The kernel sources remain closed — even though Apple promised to maintain XNU, the OS X Kernel, as open source, it apparently frees itself from that pledge for its mobile version. Occasionally, some of the iOS modifications leak into the publicly available sources (as can be seen by various `#ifdef`, `__arm__`, and `ARM_ARCH` conditionals), though these generally diminish in number with new kernel versions.

- The kernel is compiled slightly differently, with a focus on embedded features and some new APIs, some of which eventually make it to OS X, whereas others do not.
- The system GUI is Springboard, the familiar touch-based application launcher, rather than Aqua, which is mouse-driven and designed for windowing. SpringBoard proved so popular it has actually been (somewhat) back ported into OS X with Lion's LaunchPad.
- Memory management is much tighter, as there is no nigh-infinite swap space to fall on. As a consequence, programmers have to adapt to harsher memory restrictions and changes in the programming model.
- The system is hardened, or "jailed," so as not to allow any access to the underlying UNIX APIs (i.e. Darwin), nor root access, nor any access to any directory but the application's own. Only Apple's applications enjoy the full power of the system. App Store apps are restricted and subject to Apple's scrutiny.

The last point is really the most important: Apple has done its utmost to keep iOS closed, as a specialized operating system for its mobile platforms. In effect, this strips down the operating system to allow developers only the functionality Apple deems as "safe" or "recommended," rather than allow full use of the hardware, which — by itself — is comparable to any decent desktop computer. But these limitations are artificial — at its core, iOS can do nearly everything that OS X can. It doesn't make sense to write an OS from scratch when a good one already exists and can simply be ported. What's more, OS X had already been ported once, from PPC to x86 — and, by induction, could be ported again.

Whether or not you possess an i-Device, you have no doubt heard the much active buzz around the "jailbreaking" procedure, which allows you to overcome the Apple-imposed limitations. Without getting into the legal implications of the procedure (some claim Apple employs more lawyers than programmers), suffice it to say it is possible and has been demonstrated (and often made public) for all i-Devices, from the very first iPhone to the iPhone 4S. Apple seems to be playing a game of cat and mouse with the jailbreakers, stepping up the challenge considerably from version to version, yet there's always "one more thing" that the hackers find, much to Apple's chagrin.

Most of the examples shown in this book, when applied to iOS, require a jailbroken device. Alternatively, you can obtain an iOS software update — which is normally encrypted to prevent any prying eyes such as yours — but can easily be decrypted with well-known decryption keys obtained from certain iPhone-dedicated Wiki sites. Decrypting the iOS image enables you to peek at the file system and inspect all the files, but not run any processes for yourself. For this reason, jailbreaking proves more advantageous. Jailbreaking is about as harmful (if you ask Apple) as open source is bad for your health (if you ask Microsoft). Apple went so far as to "get the facts" and published HT3743[4] about the terrible consequences of "unauthorized modification of iOS." This book will not teach you how to jailbreak, but many a website will happily share this information.

If you were to, say, jailbreak your device, the procedure would install an alternate software package called Cydia, with which you can install third-party apps, that are not App Store approved. While there are many, the ones you'll need to follow along with the examples in this book are:

- **OpenSSH**: Allows you to connect to your device remotely, via the SSH protocol, from any client, OS X, Linux (wherein ssh is a native command line app), or Windows (which has a plethora of SSH clients — for example, PuTTY).

- **Core Utilities:** Packaging the basic utilities you can expect to find in a UNIX `/bin` directory.
- **Adv-cmds and top:** Advanced commands, such as `ps` to view processes.

SSHing to your device, the first command to try would be the standard UNIX `uname` which you saw earlier in the context of OS X. If you try this on an iPad 2 running iOS 4.3.3, for example, you would see something similar to the following:

OUTPUT 1-2A: uname(1) on an iOS 4 iPad 2

```
root@Padishah (/) # uname -a
Darwin Padishah 11.0.0 Darwin Kernel Version 11.0.0: Wed Mar 30 18:52:42 PDT 2011;
root:xnu-1735.46~10/RELEASE_ARM_S5L8940X iPad2,3 arm K95AP Darwin
```

And on an iPod running iOS 5:, you would see the following:

OUTPUT 1-2B: uname(1) on a 4th-generation iPod running iOS 5.0

```
root@Podicum (/) # uname -a
Darwin Podicum 11.0.0 Darwin Kernel Version 11.0.0: Thu Sep 15 23:34:16 PDT 2011;
root:xnu-1878.4.43~2/RELEASE_ARM_S5L8930X iPod4,1 arm N81AP Darwin
```

So, from the kernel perspective, this is (almost) the same kernel, but the architecture is ARM. (S5L8940X is the processor on iPad, commonly known as A5, whereas S5L8930X is the one known as A4. The new iPad is reported as iPad3.1, and its processor, A5X, is identified as S5L8945X).

Table 1-1 partially maps OS X and iOS, in some of their more modern incarnations, to the respective version of XNU. As you can see, until 4.2.1, iOS was using largely the same XNU version as its corresponding OS X at the time. This made it fairly easy to reverse engineer its compiled kernel (and with a fairly large number of debug symbols still present!). With iOS 4.3, however, it has taken off in terms of kernel enhancements, leaving OS X behind. Mountain Lion seems to put OS X back in the lead, but this might very well change if and when iOS 6 comes out.

TABLE 1-1: Mapping of OS X and iOS to their corresponding kernel versions, and approximate release dates.

OPERATING SYSTEM	RELEASE DATE	KERNEL VERSION
Puma (10.1.x)	Sep 2001	201.*.*
Jaguar (10.2.x)	Aug 2002	344.*.*
Panther (10.3.x)	Oct 2003	517.*.*
Tiger (10.4.x)	April 2005	792.*.*
iOS 1.1	June 2007	933.0.0.78
Leopard (10.5.4)	October 2007	1228.5.20

OPERATING SYSTEM	RELEASE DATE	KERNEL VERSION
iOS 2.0.0	July 2008	1228.6.76
iOS 3.1.2	June 2009	1357.5.30
Snow Leopard (10.6.8)	August 2009	1504.15.3
iOS 4.2.1	November 2010	1504.58.28
iOS 4.3.1	March 2011	1735.46
Lion (10.7.0)	August 2011	1699.22.73
iOS 5	October 2011	1878.4.43
Lion (10.7.3)	February 2012	1699.24.23
iOS 5.1	March 2012	1878.11.8
Mountain Lion (DP1)	March 2012	2050.1.12

THE FUTURE OF OS X

At the time of writing, the latest publicly available Mac OS X is Lion, OS X 10.7, with Mountain Lion — OS X 10.8 — lurking in the bushes. Given that the minor version of the latter is already at 8, and the supply of felines has been exhausted, it is also likely to be the last "OS X" branded operating system (although this is, of course, a speculation).

OS X has matured over the past 10 years and has evolved into a formidable operating system. Still, from an architectural standpoint, it hasn't changed that much. The great transition (to Intel architectures) and 64-bit changes aside, the kernel has changed relatively little in the past couple of versions. What, then, may one expect from OS XI?

- **The eradication of Mach**: The Mach APIs in the kernel, on which this book will elaborate greatly, are an anachronistic remnant of the NeXTSTEP days. These APIs are largely hidden from view, with most applications using the much more popular BSD APIs. The Mach APIs are, nonetheless, critical for the system, and virtually all applications would break down if they were to be suddenly removed. Still, Mach is not only inconvenient — but also slower. As you will see, its message-passing microkernel-based architecture may be elegant, but it is hardly as effective as contemporary monolithic kernels (in fact, XNU tends toward the monolithic than the microkernel architecture, as is discussed in Chapter 8). There is much to be gained by removing Mach altogether and solidifying the kernel to be fully BSD, though this is likely to be no mere feat.
- **ELF binaries**: Another obstacle preventing Mac OS from fully joining the UN*X sorority is its insistence on the Mach-O binary format. Whereas virtually all other UN*X support ELF, OS X does not, basing its entire binary architecture on the legacy Mach-O. If Mach is removed, Mach-O will lose its raison d'etre, and the road to ELF will be paved. This, along

with the POSIX compatibility OS X already boasts, could provide both source code and binary compatibility, allowing migrating applications from Solaris, BSD, and Linux to run with no modifications.

- **ZFS**: Much criticism is pointed at HFS+, the native Mac OS file system. HFS+ is itself a patchwork over HFS, which was used in OS 8 and 9. ZFS would open up many features that HFS+ cannot. Core Storage was a giant stride forward in enabling logical volumes and multi-partition volumes, but still leaves much to be desired.
- **Merger with iOS**: At present, features are tried out in OS X, and then sometimes ported to iOS, and sometimes vice versa. For example, Launchpad and gestures, both now mainstream in Lion, originated in iOS. The two systems are very much alike in many regards, but the supported frameworks and features remain different. Lion introduced some UI concepts borrowed from iOS, and iOS 5.0 brings some frameworks ported from OS X. As mobile platforms become stronger, it is not unlikely that the two systems will eventually become closer still, paving the way for running iOS apps, for example, on OS X. Apple has already implemented an architecture translation mechanism before — with Rosetta emulating the PPC architecture on Intel.

SUMMARY

Over the years, Mac OS evolved considerably. It has turned from being the underdog of the operating system world — an OS used by a small but devoted population of die-hard fans — into a mainstream, modern, and robust OS, gaining more and more popularity. iOS, its mobile derivative, is one of the top mobile operating systems in use today.

The next chapters take you through a detailed discussion of OS X internals: Starting with the basic architecture, then diving deeper into processes, threads, debugging, and profiling.

REFERENCES

[1] Amit Singh's Technical History of Apple's Operating Systems: `http://osxbook.com/book/bonus/chapter1/pdf/macosxinternals-singh-1.pdf`

[2] ARS Technica: `http://arstechnica.com`

[3] Wikipedia's Mac OS X entry: `http://en.wikipedia.org/wiki/Mac_OS_X`

[4] "Unauthorized modification of iOS has been a major source of instability, disruption of services, and other issues": `http://support.apple.com/kb/HT3743`